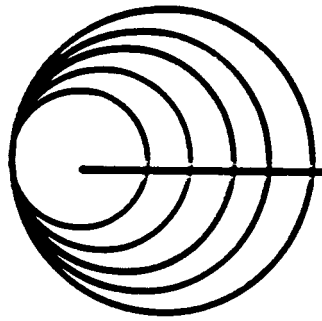
# Appendix II



# COMPUTER STUDIES

## TECHNICAL REPORT

EFFECTIVENESS OF BACK-TO-BACK TESTING

Mladen A. Vouk, David F. McAllister
David E. Eckhardt
Alper Caglayan
John P. J. Kelly

TR-87-08

**North Carolina State University**

**Raleigh, N. C. 27650**

# EFFECTIVENESS OF BACK-TO-BACK TESTING*

Mladen A. Vouk, David F. McAllister
North Carolina State University
Department of Computer Science, Box 8206, Raleigh, NC 27695-8206

David E. Eckhardt
National Aeronautics and Space Administration
Langley Research Center, Hampton, Va 23665

Alper Caglayan
Charles River Analytics
55 Wheeler St., Cambridge, Ma 02138

John P. J. Kelly
University of California, Santa Barbara
Department of Electrical and Computer Engineering, Santa Barbara, Ca 93106

**Key Words:** Software fault-tolerance, software testing, back-to-back testing, correlated errors, software reliability

## Abstract

Three models of back-to-back testing process are described. Two models treat the case where there is no inter-component failure dependence. The third model describes the more realistic case where there is correlation among the failure probabilities of the functionally equivalent components. The theory indicates that back-to-back testing can, under right conditions, provide a considerable gain in software reliability. The models are used to analyse the data obtained in a fault-tolerant software experiment. It is shown that the expected gain is indeed achieved, and exceeded, provided the inter-component failure dependence is sufficiently small. However, even with relatively high correlation the use of several functionally equivalent components coupled with the back-to-back testing may provide a considerable reliability gain. Implications of this finding are that the multiversion software development is a feasible and cost-effective approach to providing highly reliabile software components intended for fault-tolerant software systems, on condition that special attention is directed at early detection and elimination of correlated faults.

---

(*) This research was supported in part by NASA grants NAG-1-667, and NAG-1-512, and contract NAS-1-17705.

# 1. Introduction

Fault-tolerance is or will become part of many critical software and hardware systems [e.g., Mar82, Mad84, Tro85, Bis86]. There are two common methods for achieving software fault-tolerance. These are the N-version programming approach and the recovery-block approach [Ran75, Avi84].

Although existing fault-tolerant software (FTS) techniques can achieve an improvement in reliability over non-fault-tolerant software, experiments show that failure dependence among FTS system components may not be negligible in the context of current software development and testing techniques [Nag82, Sco84, Nag84, Vou85, Wig84, Kni86, Kel86]. Correlated coincidental component failures may be disastrous in current FTS approaches and can seriously undermine any reliability gains offered by the fault-tolerance mechanisms [e.g. Sco83a, Sco84, Avi84, Eck85, Vou86a]. Hence it is important to detect and eliminate them as early as possible in a FTS life-cycle.

Throughout this paper we shall use the terms "component(s)", "version(s)", "functionally equivalent software components", and "software components" interchangeably. The terms "coincident", "correlated" and "dependent" failures (faults) have the following meaning. When two or more functionally equivalent software components fail on the same input case we say that a coincident failure has occurred, and k failing components give a level-k coincident failure. The fault(s) causing a level-k failure we shall call level-k fault(s). When two or more versions give the same incorrect response, to a given tolerance, we say that an identical-and-wrong (IAW) answer was obtained. If the measured probability of the coincident failures is significantly different from what would be expected by random chance on the basis of the measured failure probabilities of the participating components [e.g. Eck85, Kni86, Vou85], then we say that the observed coincident failures are correlated or dependent, i.e. if Pr denotes probability, then

$$Pr\{ \text{version(i) fails} \mid \text{version(j) fails} \} \neq Pr\{ \text{version(i) fails} \}.$$

If a fault, or a fault combination, results in a IAW answer from k components we say that the falut(s) has (have) "span" of size k. The fault span is important because with the probability of excitation of the fault (fault intensity or visibility) it determines the level of the inter-component failure correlation for that fault.

The back-to-back testing technique discussed here involves pairwise comparison of all functionally equivalent components. Whenever a difference is observed among responses, the problem is thoroughly investigated and appropriate action is taken. If all answers are identical to within a specified tolerance then a "no detected failure" event is said to occur. We say that back-to-back testing fails when all the components fail with (within tolerance) IAW answers. Our experiments indicate that these corrrelated faults occur in practice but can be prevented or reduced.

In section 2 we present three models of the back-to-back testing process. In section 3 we use the models to analyse and discuss the experimental information concerning the effectiveness of the back-to-back testing and the multiversion development approach.

## 2. Failure Models

Our goal is to model the probability that k versions obtain IAW answers and compare it with experimental findings. We will present three models, two of which assume that the versions fail independently and the third attempts to capture the correlation between versions. We assume that m functionally equivalent software components of versions were developed by independent programming teams from equivalent specifications. We will select a subset of size k from these m versions which we will call a k-tuple.

Back-to-back testing of k components fails to signal a potential error when all

the components agree, to a given tolerance, on a value which is a wrong answer. This results in an "undetected" failure. Of course, if k=1 i.e. single component, then every test case (run time errors resulting in operating system intervention excepted) is a potential "undetected" failure in the absence of an oracle or "golden" program. In the following text the term "agreement" means equality between two responses (answers) within a tolerance TOL. It is also assumed that a "golden" or oracle answer is available.

Consider a k-tuple of components. The following two events are independent of the golden program (see examples in Figure 1).

* If all k components agree on an answer, a "COLLECTIVE AGREEMENT" event occurs.
* If there is any disagreement among the components (components being compared pairwise with each other, C(k,2) comparisons in all), a "COLLECTIVE WARNING" event occurs.

The following event depends on the golden answer:

* If all k components agree with the "golden" answer a "SUCCESS" event occurs.

The following three events are called FAILURE events and they also depend on the golden answer:

* If one or more of the component answers disagree with the golden answer a "ONEPLUS FAILURE" event occurs. This is a pessimistic (conservative) view of the failure recognition process, since one or more failures is considered to fail the k-tuple.

The following events are subevents of the ONEPLUS FAILURE event:

* If the majority of the components disagree with the golden answer then we say that a "MAJORITY FAILURE" event has occurred. The majority of k components is defined as $\lceil (k+1)/2 \rceil$. It is possible to define other intermediate states such as the majority

of components agreeing with the golden answer, two-or-more disagreeing etc.

- If all the components disagree with the golden answer then we say that an "ALL FAILURE" event takes place.

Other measure of "distance", such as the difference between the mean value of the component answers and the golden answer, may coalesce the ONEPLUS through ALL FAILURE events into a single event.

Combinations of the above "elementary" events produce the following mutually exclusive and collectively exhaustive back-to-back testing events (see Figure 1):

* If a SUCCESS occurs together with a COLLECTIVE AGREEMENT then an "OK" event occurs.
* If a SUCCESS occurs together with a COLLECTIVE WARNING then a "FALSE FLAG" event occurs. The back-to-back testing signals an error when one is not present. We note that $|a-b| \leq TOL$ and $|b-c| \leq TOL$ does not imply that $|a-c| \leq TOL$. Hence, FALSE FLAG events are not inconsistent.
* If a FAILURE occurs together with a COLLECTIVE WARNING we say that a "FLAG" event has occurred. The back-to-back testing correctly detected a potential failure (fault).
* If FAILURE occurs simultaneously with a COLLECTIVE AGREEMENT then we say that a NO_FLAG event occurs. This is the most significant back-to-back testing event. A potential failure exists (the failure is fully confirmed if ALL FAILURE has occured) but was not detected by back-to-back testing.

## 2.1 IAW Models

Formally, the probability that a given subset of k versions or a k-tuple obtains a IAW answer can be written as a conditional probability as follows. Let A denote the event "k versions obtain identical answers" (COLLECTIVE AGREEMENT),

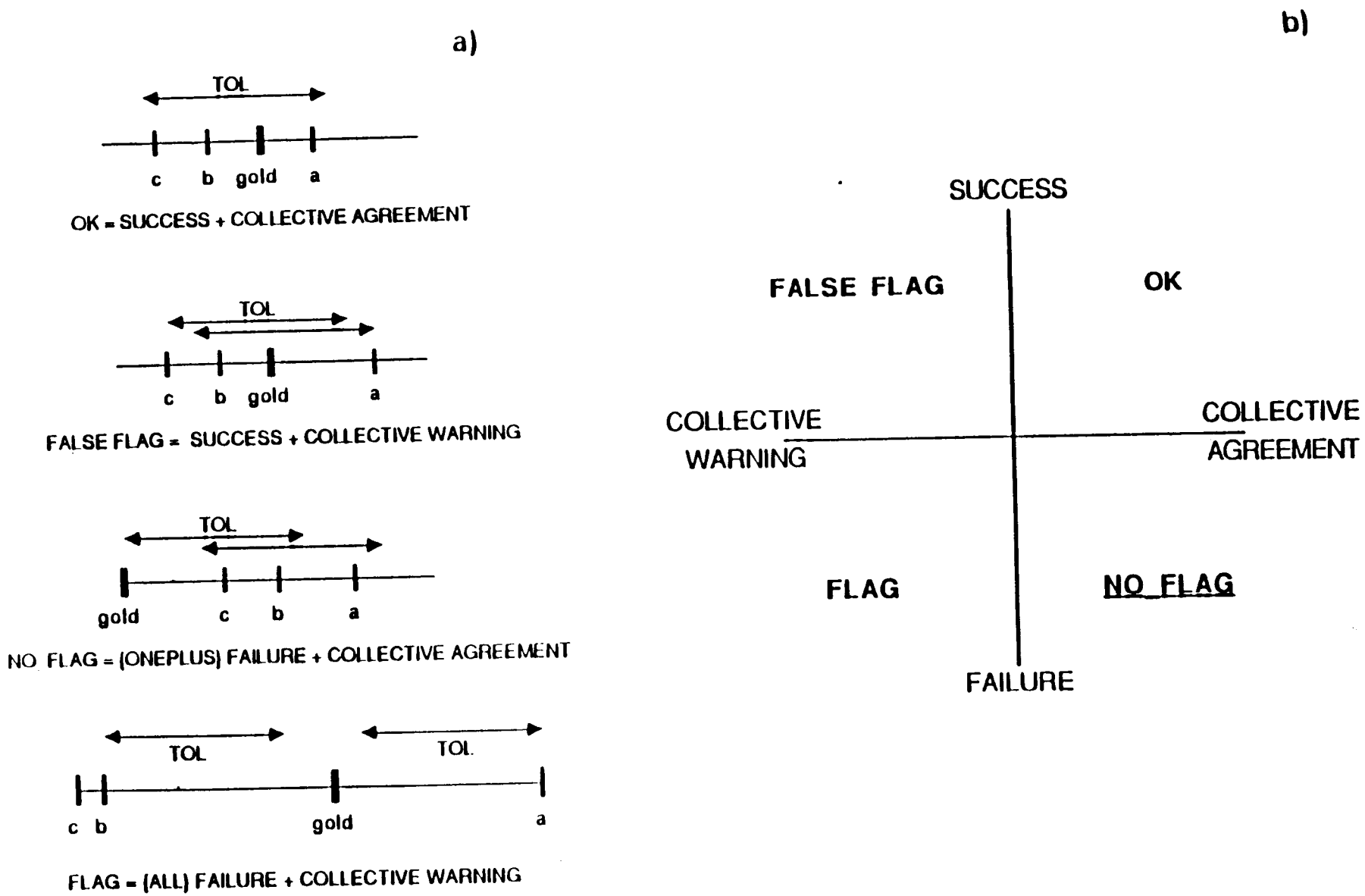



**Figure 1.** An illustration of the relationship between the tolerance TOL and elementary back-to-back testing events (a), and of the back-to-back event space (b).

an B the event "k versions fail simultaneously" (ALL FAILURE), and A&B their intersection (NO_FLAG), then

$$P(A\&B) = P(A|B)P(B) \tag{1}$$

Let $p_i$ represent the probability that component i, $1 \leq i \leq m$, fails on a given input, and let $\bar{p}$ be the mean failure probability per test case per component for the set of m components. Then

$$\bar{p} = (\sum p_i)/m \tag{2}$$

where the sum is from i=1 to m.

It is possible to construct C(m,k) sets of k-tuples from a pool of m components, where C(m,k) is the number of combinations of m objects taken k at a time. If the failure probabilities are independent then the probability of an ALL FAILURE for the $j^{th}$ k-tuple is as follows:

$$P_j(k) = \prod p_r = p_1 p_2 \cdots p_k \tag{3}$$

We will use P(k) to denote the average of the $P_j(k)$'s over all C(m,k) subsets:

$$P(k) = [\sum P_j(k)]/C(m,k) \tag{4}$$

where sum is from j=1 to C(m,k). We note that it can be shown that $(\bar{p})^k \geq P(k)$. We also note that if $\bar{p}_k$ denotes the average failure probability of a single k-tuple, then the average of this value over C(m,k) k-tuples is $\bar{p}$. In the following text, unless stated otherwise, all the quantities are averaged over C(m,k) k-tuples.

From the definition of the NO_FLAG event it follows that the probability of this event is less than or equal to the probability of a FAILURE event. Let $P_I(k)$ be the probability of an IAW answer from all k components of a k-tuple (ALL FAILURE event). Then

$$P_I(k) = \sqrt{(k)}Pr(k \text{ versions fail simultaneously}) \tag{5}$$

where $\gamma(k)$ is the conditional probability of an identical level-k answer (given an ALL FAILURE event occurred). This quantit y has , in general, two components. One is due to the cardinality of the output space, and the second component is the failure (fault) dependence. The probability of IAW answers increases as the cardinality of the output space decreases. For low cardinality (finite) output spaces the probability of a coincident failure of two or more components resulting in an IAW answer may be quite high without any correlation being present. For example, if output space is binary, then all programs which are incorrect will produce the same wrong answer, i.e. the probability of IAW answers is 1 for failing versions.

In our first model we approximate the probability of event B in equation (1) by the relationship Pr(k versions fail simultaneously) = $(\bar{p})^k$ . Therefore in Model I the probability that back-to-back testing fails to detect an error (NO_FLAG event) is

$$P_I(k) = \gamma_S(k)(\bar{p})^k \qquad (6)$$

where $\gamma_S(k)$ represents the component of $\gamma(k)$ associated with the output space cardinality effect. Since independence is assumed the failure (fault) correlation is zero. The space cardinality component $\gamma_S(k)$ is expected to be a decreasing function of the size of the error output space, x, and the number, k, of the interacting components [Sun85]. Hence, a shape similar to $1/x^k$ would be expected. In practice $\gamma_S(k)$ will also reflect the sampling strategy over the input/output domains, and will be a composite function over all the variables involved in determining the correctness of an answer. See [Sun85] for a more detailed discussion of this phenomenon.

Equations (1), (3) and (4) yield Model II for the failure probability of the back-to-back testing approach:

$$P_I(k) = \gamma_S(k)P(k) \qquad (7)$$

When $p_i$'s are equal to, say p, for all i, then the two models become equivalent, i.e. $P_I(k) = \gamma_S(k)\, p^k$ . The difference in the estimates offered by the two models depends on the variance of $\bar{p}$. It can be shown that Model I will always offer a more conservative estimate of the back-to-back failure probability than Model II. Since $\gamma_S(k) \leq 1$, P(k) and $(\bar{p})^k$ provide upper bounds or worst-case values for $P_I(k)$ .

As an illustration of the detrimental influence of inter-component failure correlation consider the following. Let $P_c(k)$ denote the average probability of an ALL FAILURE event in an environment where inter-component failure (fault) correlation is present. Then Model III is given by:

$$P_I(k) = \gamma(k) P_c(k) \tag{8}$$

The components of $\gamma(k)$ are

$$\gamma(k) = \gamma_S(k) + \gamma_c(k) - P(s\&c) \tag{9}$$

where $\gamma_c$ denotes the influence of the fault correlation, and P(s&c) the probability of the intersection of the space and correlation events.

The conditional probability $\gamma_c(k)$ is a function of the number of components containing the fault(s) resulting in an IAW answer (fault span), the visibility (or excitation probability under given sampling conditions, [Ram82]) of the fault(s), and of the number of such faults in the set under consideration.

Let $\gamma_S(k)=0$. This is often a reasonable assumption. Also assume that all the failures are caused by the same fault, or fault combination, and that all failures result in IAW answers from s components, i.e. the fault span is s. Then, given an input which results in failure, and provided $k \leq s$, we can construct C(s,k) k-tuples where all components fail with an IAW answer, and C(m,k) k-tuples in all. Therefore the probability of randomly choosing a k-tuple exhibiting level-k IAW is C(s,k)/C(m,k). If we assume that the probability of failure on input is $\bar{p}$ on the

average, then the probability that back-to-back testing fails to signal a level-k failure is

$$P_I(k) = [C(s,k)/C(m,k)]\,\bar{p} \qquad (10)$$

When $s<k_j$ $C(s,k)$ is defined to be zero. Note that if failures are completely uncorrelated fault span is one (s=1), and then $P_I(k>1)=0$

## 3. Experimental Results

In the summer 1985 a FTS experiment took place sponsored by NASA Langley Research Center. The participants were the authors, the Research Triangle Institute (NC), the University of Illinois (Urbana-Champagne, Il), and the University of Virginia (Charlottesville, Va). A detailed description of the experiment is given in [Kel86]. The programmers worked in two-person teams formed by random selection. All the programmers worked from the same specification. The programming teams were responsible for the software design, the implementation and the testing phases of the life-cycle. The experimenters provided acceptance testing of the product.

The experiment resulted in 20 functionally equivalent programs for solving a problem in inertial navigation. The problem specification was new, written for the experiment, and was not debugged via a "pilot" version of the code prior to the production of the redundant components . This resulted in a very heavy query traffic between the experimenters and the programming teams during the component design phase and in the initial stages of the implementation.

The acceptance testing was a low-expectation process, i.e. only a few critical variables were checked, and only 50 random test cases were used. Consequently the functional and structural test coverage of the products was low. The reliability of the components based on the acceptance testing was about 0.94.

The validation testing using much stricter criteria, a range of tolerances for comparing real number values, test data sets consisting of random and extremal and special value test cases, and providing full functional and linear-block coverage, detected a number of faults of varying prevalence and seriousness. Some of the faults were found to be highly correlated. The reliability of the components was found to be a strong function of the tolerance used for comparisons. Adjudication of the answer correctness was performed using a "golden" or oracle program developed at NCSU. Software development and testing was done on VAX 11/750 and 780 hardware running UNIX 4.2BSD, and MicroVAX II hardware running Ultrix 1.2.

In order to study the influence of component reliability and inter-component failure dependence on the performance of back-to-back testing we have formed subsets of components. The subsets had different average component failure probability ($\bar{p}$), and different inter-component correlation characteristics. Components for the subsets were selected on the basis of their behaviour during different stages of the validation testing. The four subsets which are discussed in this paper are coded 6(2.1), 4(2.1), 9(2.1) and 13(3.1). The first number identifies the number of versions (m) and the second the problem specification update number to which the test data and the golden code used for testing conformed.

## 3.2 Experimental Measurements

The effectiveness of back-to-back testing was investigated using random test cases. The error detecting power, and the structural and functional coverage provided by the random sets saturated very rapidly. Measured values (e.g. $\hat{\gamma}(k)$) stabilized by the time about 100 cases were run (not an unexpected result, [Vou86a,b]), and hence we used only 200 random test cases. In the back-to-back event space this, of course, expands to 200*C(m,k) event samples and gives acceptable 95% confidence bounds on the back-to-back testing parameters.

To measure the "reliability gain" (or unreliability reduction factor), G(k), offered by back-to-back testing process of k components, as opposed to the development of a single component, we shall use the ratio of the probability of an "undetected" failure in an average single component to the probability of an "undetected" failure in an average k-tuple after back-to-back testing:

$$G(k) = \overline{p}/P_I(k) \qquad (11)$$

Experimentally, single component "undetected" failures were recorded by running functionally equivalent components against a "golden" or oracle program to estimate component failure probabilities. An average value was then computed for the pool of available (operational) components.

The "undetected" multicomponent failures probabilities were computed from pairwise comparisons of responses of all components. The results of the comparisons for each test case were recorded in a (k+1) by k response matrix. The zeroth row of the response matrix (i=0) contains information on the comparison of the components with the golden code. The remaining rows cary information about the mutual comparisons of the components. For example, if the comparison of components i and j detected a difference for a given test case then the entries (i,j) and (j,i) were given value 1, otherwise the value was zero. Unless stated otherwise, a comparison involved eleven variables, or 52 individual values if array elements are counted separately. A difference was signalled if even one of these 52 values differed from the golden value. The results shown in this paper were obtained with TOL=0.0001 absolute for real numbers, and TOL=0 for integers. The response matrices were used to compute the usual multiple component failure profiles [Vou85,86a], intensity profiles [Eck85], and counts of the back-to-back events (see section 2).

Let a hat, $\hat{}$ , denote experimentally obtained estimates. If $\hat{p}_i$ denotes an

estimate of the failure probability of component i (relative to the gold program), then $\hat{\bar{p}}$ was computed by substitution of the $\hat{p}_i$ values into equation (2). Individual $\hat{P}_j(k)$'s were similarly computed by substitution in equation (3), and $\hat{P}(k)$ was then computed using equation (4). The estimate $\hat{P}_c(k)$ was computed from the count of all level-k FAILURE events. The estimate $\hat{P}_I(k)$ was calculated from the ratio [NO_FLAG-count/200*C(m,k)], where the count was over all 200 test cases and over all the k_out_of_m possible k-tuples. The parameter $\gamma$ was estimated from the ratio [NO_FLAG-count/FAILURE-count]. The analysis was performed for all three FAILURE event categories defined in section 2, i.e. ONEPLUS, MAJORITY and ALL. The results are summarized in Table 1.

To illustrate the relative size of the inter-component correlation among the sets, and order them by correlation level, we compute a function L(k) defined by:

$$L(k) = \hat{P}_I(k)/\hat{P}(k) \qquad (12)$$

where L(k) may be regarded as the amplification factor of the worst-case uncorrelated back-to-back testing failure probability required to achieve the observed $\hat{P}_I(k)$ . The value of L(k) is always positive and may be larger than 1. Since the output space cardinality is the same for all subsets any differences in the L(k)'s stem from the inter-component failure dependence and therefore can be used to estimate its relative magnitude.

### 3.3 The Gain

The experimental gain estimates (using ONEPLUS FAILURE events) are shown in Figure 2. Note that the ordinate uses logarithmic scale. The notation used in the legend of this and other figures has the following meaning. The first two letters describe the function that is being plotted. If the first letter is T then the curve is the result of theoretical computations, if it is E the data was obtained

# Table 1.

## Experimental Results

| k | $\hat{G}(k)$ | $\hat{P}_I(k)$ | $\hat{P}_C(k)$ | $\hat{P}(k)$ | $\hat{\gamma}(k)$ | sample size |
|---|---|---|---|---|---|---|
| **Set:** | **6(2.1)** | $\bar{p}$ = 0.379 using ONEPLUS FAILURE events | | | | |
| 2 | 2.80 | 0.136 | 0.615 | 0.126 | 0.221 | 3000 |
| 3 | 9.85 | 0.0385 | 0.769 | 0.0351 | 0.0501 | 4000 |
| 4 | 51.7 | 7.33e-3 | 0.873 | 7.85e-3 | 8.40e-3 | 3000 |
| 5 | 455.0 | 8.33e-4 | 0.947 | 1.38e-3 | 8.80e-4 | 1200 |
| 6 | inf | 0 | 1.0 | 2.01e-4 | 0 | 200 |
| **Set:** | **6(2.1)** | $\bar{p}$ = 0.379 using MAJORITY FAILURE events | | | | |
| 2 | 12.5 | 0.0303 | 0.144 | 0.126 | 0.211 | 3000 |
| 3 | 24.1 | 0.0156 | 0.306 | 0.0351 | 0.0515 | 4000 |
| 4 | 284.3 | 1.33e-3 | 0.156 | 7.85e-3 | 8.55e-3 | 3000 |
| 5 | inf | 0 | 0.270 | 1.38e-3 | 0 | 1200 |
| 6 | inf | 0 | 0.160 | 2.02e-4 | 0 | 200 |
| **Set:** | **6(2.1)** | $\bar{p}$ = 0.379 using ALL FAILURE events | | | | |
| 2 | 12.5 | 0.0303 | 0.144 | 0.126 | 0.211 | 3000 |
| 3 | 137.8 | 2.75e-3 | 0.0625 | 0.0351 | 0.0440 | 4000 |
| 4 | 1137.1 | 3.33e-4 | 0.0313 | 7.85e-3 | 0.0106 | 3000 |
| 5 | inf | 0 | 0.0192 | 1.38e-3 | 0 | 1200 |
| 6 | inf | 0 | 0.0150 | 2.02e-4 | 0 | 200 |
| **Set:** | **4(2.1)** | $\bar{p}$ = 0.185 using ONEPLUS FAILURE events | | | | |
| 2 | 1.12 | 0.166 | 0.302 | 0.0276 | 0.550 | 1200 |
| 3 | 1.66 | 0.111 | 0.376 | 3.31e-3 | 0.296 | 800 |
| 4 | 3.08 | 0.060 | 0.420 | 3.44e-4 | 0.143 | 200 |
| **Set:** | **4(2.1)** | $\bar{p}$ = 0.185 using MAJORITY FAILURE events | | | | |
| 2 | 4.19 | 0.0442 | 0.0683 | 0.0276 | 0.646 | 1200 |
| 3 | 3.61 | 0.0513 | 0.153 | 3.31e-3 | 0.336 | 800 |
| 4 | 18.5 | 0.010 | 0.060 | 3.44e-4 | 0.167 | 200 |
| **Set:** | **4(2.1)** | $\bar{p}$ = 0.185 using ALL FAILURE events | | | | |
| 2 | 4.19 | 0.0442 | 0.0683 | 0.0276 | 0.646 | 1200 |
| 3 | 29.6 | 6.25e-3 | 0.0262 | 3.31e-3 | 0.238 | 800 |
| 4 | inf | 0 | 0.0150 | 3.44e-4 | 0 | 200 |

Table 1. (continued)

| k | $\hat{G}(k)$ | $\hat{P}_I(k)$ | $\hat{P}_C(k)$ | $\hat{P}(k)$ | $\hat{\gamma}(k)$ | sample size |
|---|---|---|---|---|---|---|
| Set: | 9(2.1) | $\bar{p}$ = 0.366 using ONEPLUS FAILURE events | | | | |
| 2 | 2.11 | 0.174 | 0.562 | 0.126 | 0.309 | 7200 |
| 3 | 4.40 | 0.0832 | 0.683 | 0.0406 | 0.122 | 16800 |
| 4 | 9.74 | 0.0376 | 0.767 | 0.0122 | 0.0490 | 25200 |
| 5 | 23.4 | 0.0157 | 0.830 | 3.41e-3 | 0.0189 | 25200 |
| 6 | 60.9 | 6.01e-3 | 0.882 | 8.71e-4 | 6.82e-3 | 16800 |
| 7 | 175.7 | 2.08e-3 | 0.926 | 2.00e-4 | 2.25e-3 | 7200 |
| 8 | 658.6 | 5.56e-4 | 0.964 | 4.05e-5 | 5.76e-4 | 1800 |
| 9 | inf | 0 | 1.000 | 7.25e-6 | 0 | 200 |
| Set: | 9(2.1) | $\bar{p}$ = 0.366 using ALL FAILURE events | | | | |
| 2 | 6.72 | 0.054 | 0.169 | 0.126 | 0.322 | 7200 |
| 3 | 26.6 | 0.0137 | 0.0936 | 0.0406 | 0.147 | 16800 |
| 4 | 93.2 | 3.93e-3 | 0.0556 | 0.0122 | 0.0706 | 25200 |
| 5 | 279.5 | 1.31e-3 | 0.0347 | 3.41e-3 | 0.0377 | 25200 |
| 6 | 768.6 | 4.76e-4 | 0.0226 | 8.71e-4 | 0.0211 | 16800 |
| 7 | 2635.2 | 1.39e-4 | 0.0155 | 2.00e-4 | 8.93e-3 | 7200 |
| 8 | inf | 0 | 0.0117 | 4.05e-5 | 0 | 1800 |
| 9 | inf | 0 | 0.0100 | 7.25e-6 | 0 | 200 |
| Set: | 13(3.1) | $\bar{p}$ = 0.443 using ONEPLUS FAILURE events | | | | |
| 2 | 2.45 | 0.181 | 0.631 | 0.189 | 0.286 | 15600 |
| 3 | 6.53 | 0.0656 | 0.731 | 0.0782 | 0.0697 | 57200 |
| 4 | 18.8 | 0.0232 | 0.789 | 0.0311 | 0.0294 | 143000 |
| 5 | 56.0 | 7.76e-3 | 0.821 | 0.0119 | 9.46e-3 | 257400 |
| 6 | 187.1 | 2.32e-3 | 0.839 | 4.38e-3 | 2.77e-3 | 343200 |
| 7 | 739.1 | 5.86e-4 | 0.848 | 1.55e-3 | 6.94e-4 | 343200 |
| 8 | 3732.3 | 1.17e-4 | 0.853 | 5.27e-4 | 1.37e-4 | 275400 |
| 9 | 31102 | 1.40e-5 | 0.854 | 1.73e-4 | 1.64e-5 | 143000 |
| 10 | inf | 0 | 0.855 | 5.46e-5 | 0 | 57200 |
| 11 | inf | 0 | | 1.67e-5 | 0 | 15600 |
| 12 | inf | 0 | | 4.94e-6 | 0 | 2600 |
| 13 | inf | 0 | | 1.42e-6 | 0 | 200 |
| Set: | 13(3.1) | $\bar{p}$ = 0.443 using ALL FAILURE events | | | | |
| 2 | 4.82 | 0.0920 | 0.254 | 0.189 | 0.362 | 15600 |
| 3 | 22.2 | 0.0199 | 0.167 | 0.0782 | 0.119 | 57200 |
| 4 | 86.3 | 5.13e-3 | 0.123 | 0.0311 | 0.0418 | 143000 |
| 5 | 304.1 | 1.46e-3 | 0.0976 | 0.0119 | 0.0149 | 257400 |
| 6 | 1187.8 | 3.73e-4 | 0.0810 | 4.38e-3 | 4.61e-3 | 343200 |
| 7 | 6081.5 | 7.28e-5 | 0.0686 | 1.55e-3 | 1.06e-3 | 343200 |
| 8 | 57014 | 7.77e-6 | 0.0587 | 5.27e-4 | 1.32e-4 | 275400 |
| 9 | inf | 0 | 0.0501 | 1.73e-4 | 0 | 14300 |
| 10 | inf | 0 | 0.0422 | 5.46e-5 | 0 | 57200 |
| 11 | inf | 0 | 0.0347 | 1.67e-5 | 0 | 15600 |
| 12 | inf | 0 | 0.0273 | 4.94e-6 | 0 | 2600 |
| 13 | inf | 0 | 0.0200 | 1.42e-6 | 0 | 200 |

experimentally. The second letter has the following meanings: G = G(k) or gain, L = L(k), c = $\gamma$(k). The number following the first two letters denotes the number of versions involved in the comparisons (m), and is used to identify the component subset used. If the data are experimental this number may be followed by another letter. Letter A denotes that ALL FAILURE events were used to derive the plotted values, letter M that the MAJORITY FAILURE events were used, and if there is no letter ONEPLUS events were used. For theoretical curves the number of components is followed, in parentheses, by a roman numeral (I, II or III) identifying the theoretical model bound used to compute the values. In the case of Model III the identifier is followed by the span value used in computations.

It is obvious that even in the worst observed case (subset 4(2.1)) the multiversion development coupled with back-to-back testing offers some gain in reliability over the single component development approach. The size of the fault correlation level, as measured by L(k), is illustrated in Figure 3. Experimental $\hat{\gamma}$(k) estimates are shown in figure 4. The largest inter-component fault-correlation is exhibited by set 4(2.1) and the smallest by set 6(2.1). From Table 1 we see that the most unreliable set is 13 (average failure probability is 0.443), and the most reliable subset is 4(2.1) with an average failure probability of 0.185. The component sets 6, 13 and 9 reach infinite gain (no "undetected" failures (faults)) for 6, 10 and 9 developed components respectively. Using the conservative ONEPLUS FAILURE events, subset 4(2.1) never detects all the potential failures.

The slopes of the curves in Figure 2, and the gain they imply vary among the subsets. The reason for this difference is primarily the inter-component failure correlation. The influence of the average component failure probability of a set appears to be far less important than the correlation effect. For example, the sets 6 and 9 are approximately equally reliable but the lower correlation set 6(2.1) offers

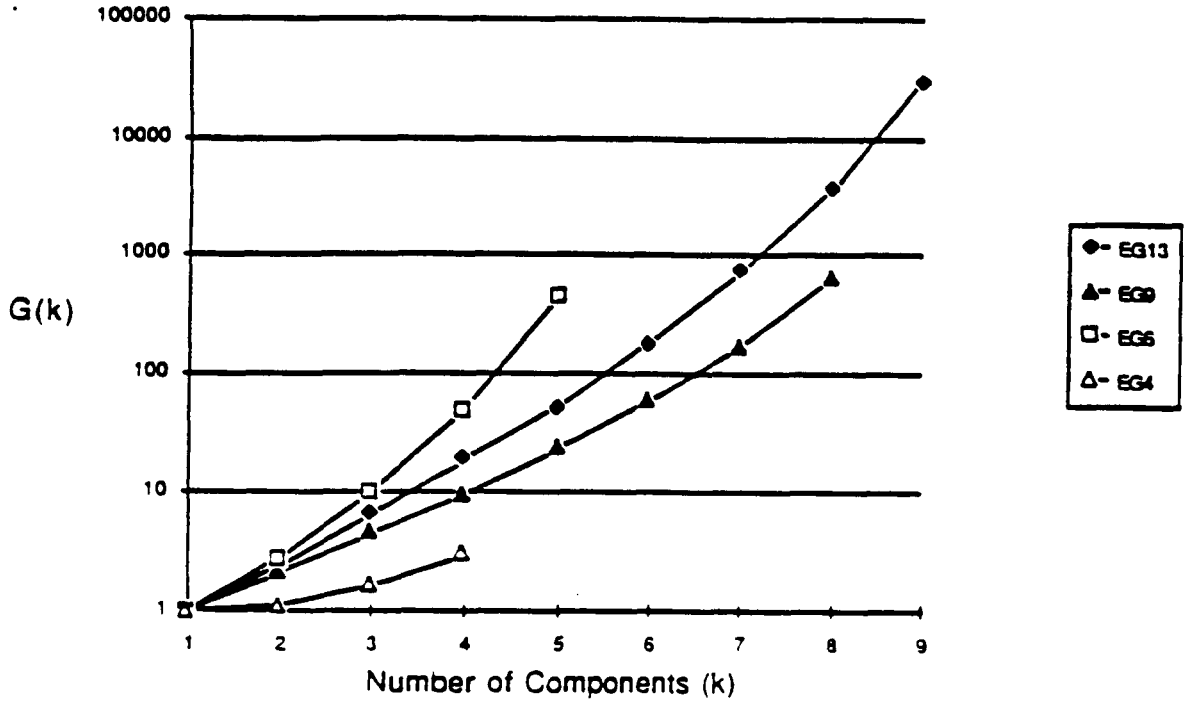



Figure 2. Gain, G(k), vs. Number of Developed components (k). The gain estimate of the ratio of "undetected" failures in an average single component to "undetected" failures remaining after back-to-back testing of the components computed using ONEPLUS FAILURE events.

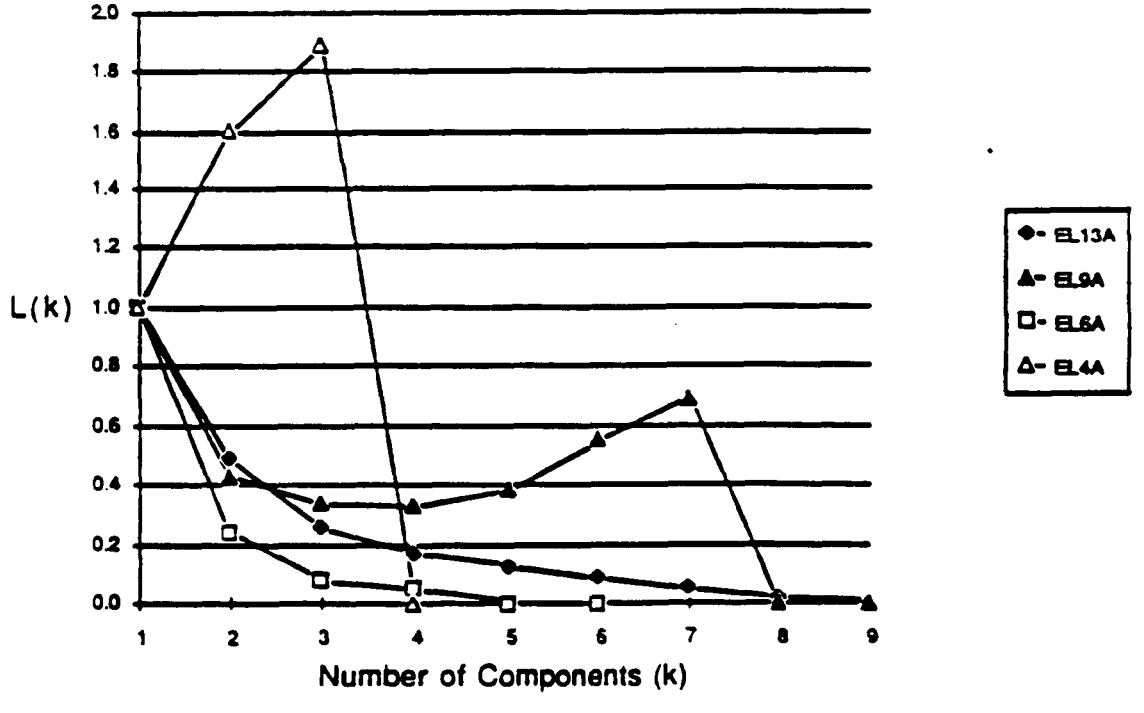



Figure 3. L(k) vs. Number of components(k). Illustration of the relative inter-component correlation. The difference between the curves indicates the difference in the failure (fault) dependence.

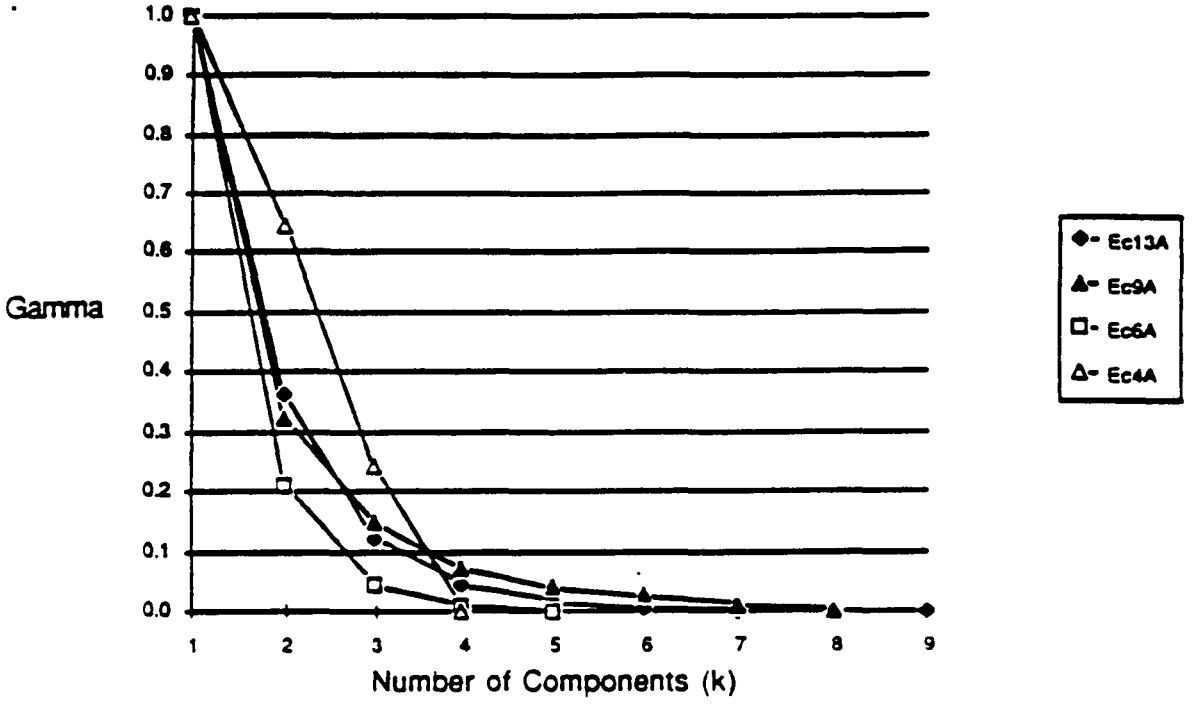



**Figure 4.** $\gamma(k)$ vs. Number of Components. Experimental estimate $\hat{\gamma}(k)$ computed using ALL FAILURE events.

better gain figures. Similarly, the most "unreliable" set, 13(2.1), has a correlation level which appears to be smaller than that of set 9(2.1), and its gain curve lies above that for the set 9. On the other hand, set 4(2.1) has relatively high reliability, but its components are highly correlated resulting in a gain curve far below any of the other sets.

Figures 5 to 8 show the experimental and theoretical gain curves for each of the component subsets separately. Filled (black) symbols refer to the experimental data and unfilled symbols to theoretical computations. Theoretical computations represent worst-case bounds obtained using Model I (triangles), Model II (squares), and Model III (diamonds, equation (10) using the maximum fault span observed for

conservative gain estimates). In the case of set 4(2.1) this last limit would be constant and equal to one, so diamonds in that case represent computations for the span of 3 recorded using the ALL FAILURE events.

It should be noted that the theoretical models, as defined in section 2, do not account for the tolerance effect (i.e. a range of FAILURE events from ONEPLUS to ALL), but only for the ALL FAILURE events. Therefore the theoretical values obtained using these models will underestimate the actual failure probability as measured by FAILURE or MAJORITY FAILURE events. Hence, to validate the models we use the ALL FAILURE event data. Also note that any result checking during the development/testing of components effectively acts as an additional version (even manual computations may qualify as a "version"). Therefore, in practice the minimal number of "developed" components is usually 2.

Figure 5 shows the FAILURE (EG6), MAJORITY FAILURE (EG6M), and ALL FAILURE (EG6A), estimates of the gain for the six component set. Also shown are the worst-case gain curves expected using Model I, TG6(I), and Model II, TG6(II), as well as a Model III based bound (equation (10) with m=6, s=5, $\bar{p}$=0.379), TG6(III/5). It is interesting to observe that for the ALL FAILURE events the maximum fault span is one less than it is for the ONEPLUS FAILURE events. The conservative experimental gain curve is well approximated by the Model II worst-case bound, while the MAJORITY and ALL FAILURE estimates are better then this bound.

Figure 6 shows the gain curves for the subset 9(2.1). Only the ONEPLUS FAILURE and the ALL FAILURE experimental data are given. The component sets 6(2.1) and 9(2.1) have very similar average component failure probabilities, but they have significantly different inter-component failure dependence characteristics (see Figure 3). The effect of the increased inter-component failure correlation in subset 9(2.1) manifests as a reduced slope of the 9 gain curves. The conservative gain

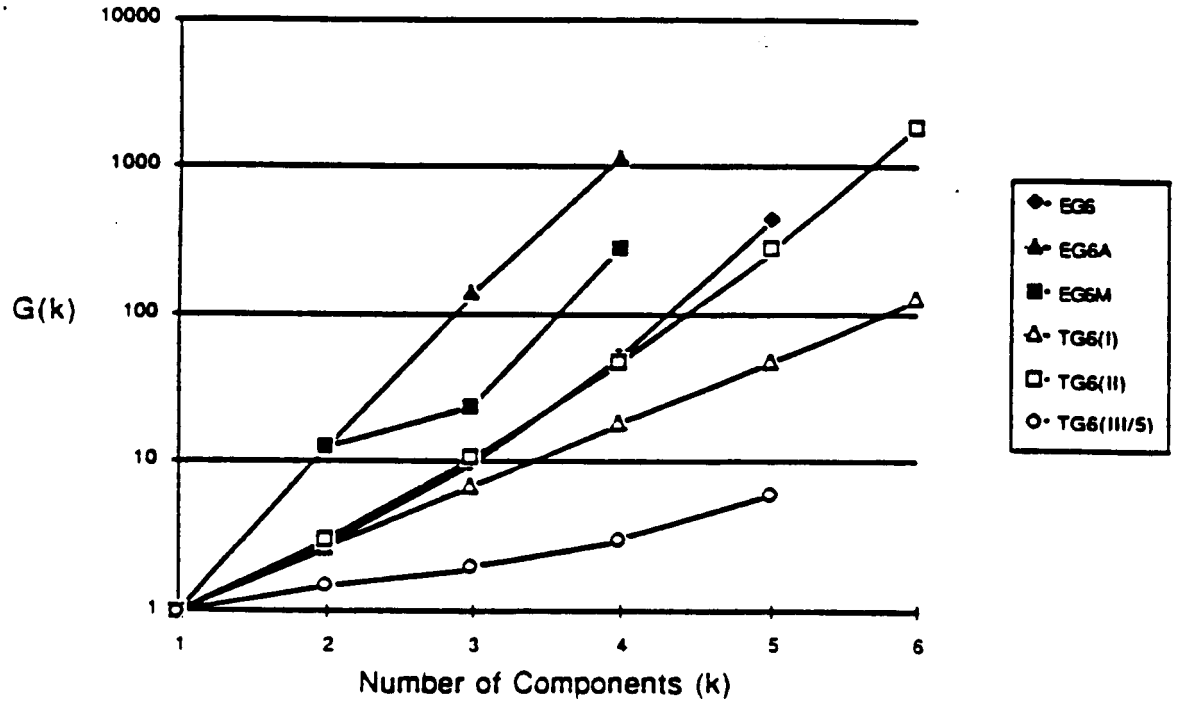



**Figure 5.** G(k) vs. Number of Components. Experimental and theoretical gain curves for set 6(2.1).

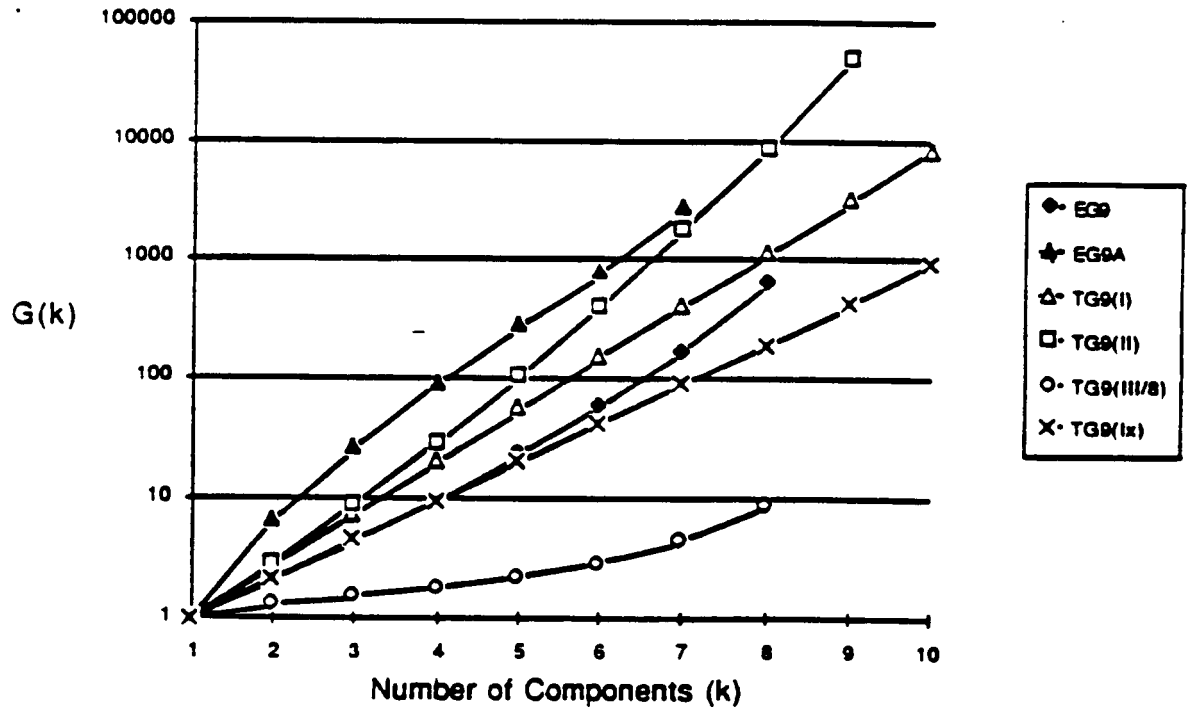



**Figure 6.** G(k) vs. Number of Components. Experimental and theoretical gain curves for set 9(2.1).

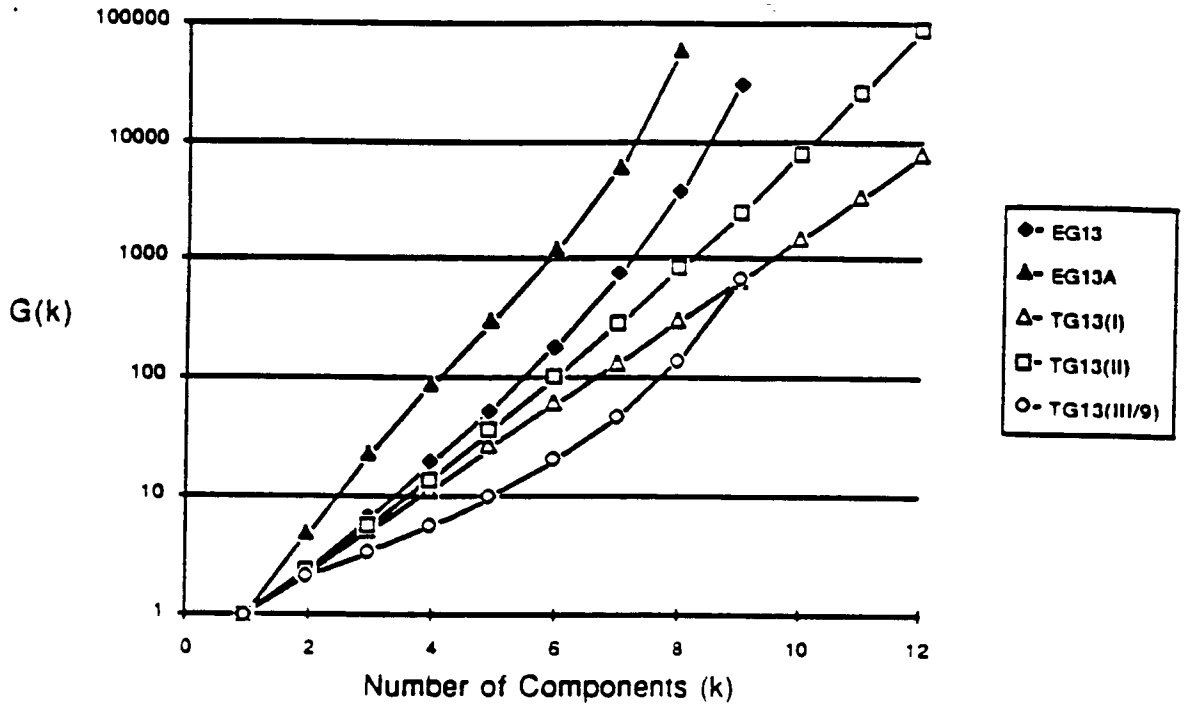



Figure 7. G(k) vs. Number of Components. Experimental and theoretical gain curves for set 13(3.1).

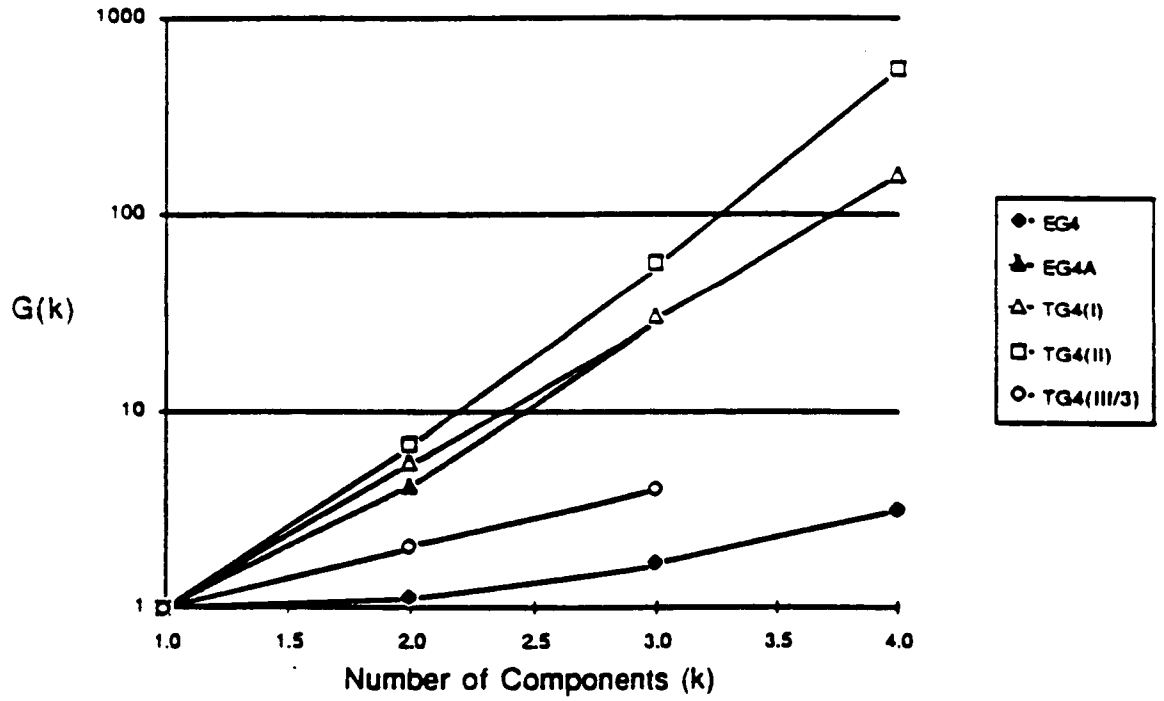



Figure 8. G(k) vs. Number of Components. Experimental and theoretical gain curves for set 4(2.1).

estimates fall below the Model I predictions based on the average failure probability of the whole subset. Figures 7 and 8 illustrate the gain information for subsets 13(3.1) and 4(2.1) respectively.

Considering all four sets we note that the Model I worst-case bound provides a satisfactory lower limit with respect to all ALL FAILURE experimental curves. A reasonable conservative limit seems to be provided through the Model III bound. Work in progress at NCSU shows that good estimates of the correlation behaviour and of the bounds can be obtained without the use of a special golden program. For example, curve TG9(Ix) was computed using Model I with an estimate of $\bar{p}$ based only on the relative performance of the 9 components. Each component was in turn treated as the gold program and average failure probability of the other components was computed relative to it. A grand average was then computed over all the estimates for us in Model I.

## 4. Conclusions

Using functionally equivalent software components we have experimentally investigated the effectiveness of back-to-back testing process. We compared the unreliability offered by a multiversion development approach with back-to-back testing, with the average unreliability of a single component. Even conservative estimates indicate a considerable increase in the probability of detecting failures (faults) if back-to-back testing is used. Three models of the back-to-back testing process were presented, and it was shown that they offer good estimates of the lower bounds on the observed multiversion development reliability gains.

## 5. References


[Avi84] A. Avizienis and J.P. Kelly, "Fault-Tolerance by Design Diversity: Concepts and Experiments", Computer, Vol. 17, pp. 67-80, 1984

[Bis86] P.G. Bishop, D.G. Esp, M. Barnes. P Humphreys, G. Dahl, and J. Lahti, "PODS—A Project on Diverse Software", IEEE Trans. Soft. Eng., Vol. SE-12(9), 929-940, 1986.

[Eck85] D.E. Eckhardt, Jr. and L.D. Lee, "A Theoretical Basis for the Analysis of Multiversion Software Subject to Coincident Errors", IEEE Trans. Soft. Eng., Vol. SE-11(12), 1511-1517, 1985.

[Kel86] J. Kelly, D. Eckhardt, A. Caglayan, J. Knight, D. McAllister, M. Vouk, "Early Results from the Second Generation Multi-Version Software Experiment", submitted for publication, 1986.

[Kni86] J.C. Knight and N.G. Leveson, "An Experimental Evaluation of the assumption of Independence in Multiversion Programming", IEEE Trans. Soft. Eng., Vol. SE-12(1), 96-109, 1986.

[Mad84] W.A. Madden, and K.Y. Rone, "Design, Development, Integration: Space Shuttle Primary Flight Software System", Comm. of the ACM, Vol. 27(8), 902-913, 1984.

[Mar82] D.J. Martin, "Dissimilar Software in High Integrity Applications in Flight Controls", Proc. AGARD - CP 330. 36.1-36.13, September 1982.

[Nag82] P.M. Nagel and J.A. Skrivan, "Software Reliability: Repetitive Run Experimentation and Modeling", BSC-40366, Boeing, Seattle. Wa., 1982

[Nag84] P.M. Nagel, F.W. Scholz and J.A. Skrivan, "Software Reliability: Additional Investigation into Modeling with Replicated Experiments", NASA CR172378, Boeing, Seattle, Wa., 1984

[Ram82] C.V. Ramamoorthy and F.B. Bastani, "Software reliability - status and perspectives", IEEE Trans. Soft. Eng., Vol. SE-8, 354-371, 1982

[Ran75] B. Randell, "System structure for software fault-tolerance", IEEE Trans. Soft. Eng., Vol. SE-1, 220-232, 1975

[Sco83,a] R.K. Scott, "Data Domain Modeling of Fault Tolerant Software Reliability", Ph.D. Dissertation, North Carolina State University, Raleigh, North Carolina, 1983

[Sco83,b] R.K. Scott, J.W. Gault, D.F. McAllister and J. Wiggs, "Experimental Validation of Six Fault-Tolerant Software Reliability Models", Proc. IEEE 14th Fault-Tolerant Computing Symposium, pp. 102-107, 1983

[Sco84] R.K. Scott, J.W. Gault, D.F. McAllister and J. Wiggs, "Investigating Version Dependence in Fault-Tolerant Software", AGARD 361, pp. 21.1-21.10, 1984

[Sco86,b] R.K. Scott, J.W. Gault and D.F. McAllister, "Fault-Tolerant Software Reliability Modeling", IEEE Trans. Software Eng., 1986, to appear

[Sun85] C. Sun, "Reliability of N-version programming for finite output spaces", M.Sc. Thesis, North Carolina State University, Raleigh, North Carolina, 1985

[Tro85] R. Troy and C. Baluteau, "Assessment of Software Quality for the Airbus A310 Automatic Pilot", Proc. FTCS 15, Ann Arbor, USA, (IEEE CS Press), 438-443, June 1985.

[Vou85] M.A. Vouk, D.F. McAllister, K.C. Tai, "Identification of correlated failures of fault-tolerant software systems", in Proc. COMPSAC 85, 437-444, 1985.

[Vou86a] M.A. Vouk, D.F. McAllister, K.C. Tai, "An Experimental Evaluation of the Effectiveness of Random Testing of Fault-tolerant Software", Proc. Workshop on Software Testing, Banff, Canada, IEEE CS Press, July 1986.

[Vou86b] M.A. Vouk, M.L. Helsabeck, K.C. Tai, and D.F. McAllister, "On Testing of Functionally Equivalent Components of Fault-Tolerant Software", Proc. COMPSAC 86, 414-419, 1986.

[Wig84] J.E. Wiggs, "Experimental Validation of Fault-Tolerant Software Reliability Models", M.Sc. Thesis, North Carolina State University, Raleigh, North Carolina, 1984